Facultad de Ciencias

CLASIFICACIÓN DE IMÁGENES DE ESPECIES DE ZOOPLANCTON UTILIZANDO DEEP LEARNING

(Zooplankton species image classification using Deep Learning)

Trabajo de Fin de Máster
para acceder al

MÁSTER EN DATA SCIENCE

Autor: Jaime Céspedes Sisniega

Director: Lara Lloret Iglesias

Co-Director: Ignacio Heredia Cacha

Octubre - 2018

Índice

Índice de figuras

Índice de tablas

Agradecimientos

En primer lugar, me gustaría agradecer a mi familia y amigos el apoyo recibido durante el transcurso del máster.

Dar las gracias también a Lara e Ignacio por ayudarme y darme las pautas necesarias en todo momento para poder afrontar la realización de este trabajo de fin de máster.

Por último, agradecer también al IFCA (Instituto de Física de Cantabria) el poner a disposición los recursos computacionales necesarios para realizar el proyecto.

Resumen

Hoy en día, con la continua generación de grandes volúmenes de datos propiciada por los recientes avances en aspectos tecnológicos que se vienen dando en los últimos años, son muchos los sectores e industrias que deciden sacar un rendimiento de los mismos haciendo uso de algoritmos de aprendizaje automático, con el objetivo de poder aplicar el conocimiento obtenido a una posterior toma de decisiones.

Desde el punto de vista más práctico, estos avances pueden ser utilizados para realizar tareas que requieren de un tiempo relativamente amplio para ser completadas de forma manual, como por ejemplo, el problema que se plantea en el presente trabajo, en el que se propone desarrollar un modelo que sea capaz de clasificar imágenes de especies de zooplancton en sus respectivas categorías.

Para llevar a acabo esta tarea de clasificación de imágenes, se hace uso de aprendizaje profundo o también denominado como *deep learning* a través de la utilización de redes convolucionales.

Palabras clave: Aprendizaje automático, aprendizaje profundo, redes neuronales convolucionales, redes neuronales residuales, clasificación de zooplancton

Abstract

Nowadays large volumes of data are continously generated due to the recent technological advances. Many sectors and industries decide to take advantage of them using machine learning algorithms, hoping to apply the knowledge obtained to enhance decision making.

In practice, the advances can be used to carry out tasks that require a relatively long time to be completed manually. In the present work, we develop a model that is capable of classifying images of zooplankton species in their respective categories.

We use convolutional neural networks, a particular deep learning technique, for the classification of the images.

Keywords: Machine learning, deep learning, convolutional neural networks, residual neural networks, zooplankton clasification

1. Introducción

En los últimos años, la mejora en aspectos algorítmicos y sobretodo computacionales está permitiendo el avance en campos relacionados con la inteligencia artificial, como es el aprendizaje automático. A todo ello se le suma la gran cantidad de datos que son generados continuamente, denominado como *Big Data* [6], lo que hace que deban existir herramientas y personas dispuestas a interpretar y sacar el mayor valor posible de esta ingente cantidad de información.

Este trabajo tiene por objetivo generar modelos que sean capaces de clasificar imágenes de zooplancton en sus correspondientes categorías. Para llevar a cabo dicha tarea se hace uso de *deep learning*. Uno de los motivos que ha llevado al *deep learning* a convertirse en una técnica tan usada en los últimos años es la gran cantidad de datos que se generan a día de hoy, ya que gran parte de la base de los algoritmos usados en *deep learning* ya existían desde los años 50, como el perceptrón [7], o la posterior incursión unas décadas después de algoritmos capaces de ajustar los pesos de las redes neuronales por sí mismas, como es el caso del algoritmo *backpropagation* [8]. La generación de grandes volúmenes de datos se debe a que en la actualidad la inmensa mayoría de las acciones son llevadas a cabo con dispositivos electrónicos, desde los computadores y los dispositivos móviles, hasta las pulseras que registran la actividad, haciendo que gran parte de las empresas apuesten por el *deep learning* para darles sentido a dichos datos almacenados y poder obtener algún tipo de conocimiento de ellos. Otro de los factores que pueden ser decisivos a la hora de utilizar *deep learning* es la facilidad de obtener características relevantes de un problema por sí mismo durante el proceso de aprendizaje, siendo una etapa relativamente costosa en cuanto a términos computacionales se refiere, aunque a través del uso de las GPU's (*Graphics Processing Unit*) se puede conseguir acelerar los cálculos que son llevados a cabo, lo que implica una disminución del tiempo requerido para el aprendizaje.

Para el problema tratado se hace uso de datos procedentes de una base de datos que contiene más de 850000 imágenes de especies de zooplancton, estableciendo el tamaño total de las mismas en 7,1 GB, haciendo que se pueda considerar como un problema de *Big Data*. Para que la realización del trabajo sea posible, se han establecido una serie de tareas:

- Extracción de las imágenes de la base de datos NoSQL.
- Generación de modelos que sean capaces de clasificar las imágenes de zooplancton en sus correspondientes categorías.
- Interpretación de los resultados obtenidos y descripción de las posibles

futuras mejoras.

1.1. Motivación

En medios acuáticos podemos encontrar un amplio rango de organismos y especies, con tamaños totalmente diferentes. Este trabajo se centra en el uso de imágenes de varios de estos organismos, en concreto pertenecientes al zooplancton. Estas especies poseen un tamaño muy reducido por lo general, haciendo que su visualización en muchos casos sea posible únicamente a través de microscopio. Cabe destacar la capacidad de algunos tipos de zooplancton para la ingesta de microplásticos [9, 10] o para la medición de la calidad del agua de lagos [11]. Al igual que ocurre con el fitoplancton, puede servir de indicador del estado general del ecosistema marino. Ante la dificultad que puede acarrear la clasificación de cada una de las especies de forma manual, debido al problema comentado anteriormente del tamaño de las mismas, se plantea automatizar esta tarea haciendo uso de *deep learning*, de forma que dada una imagen, se pueda clasificarla en su correspondiente especie de zooplancton.

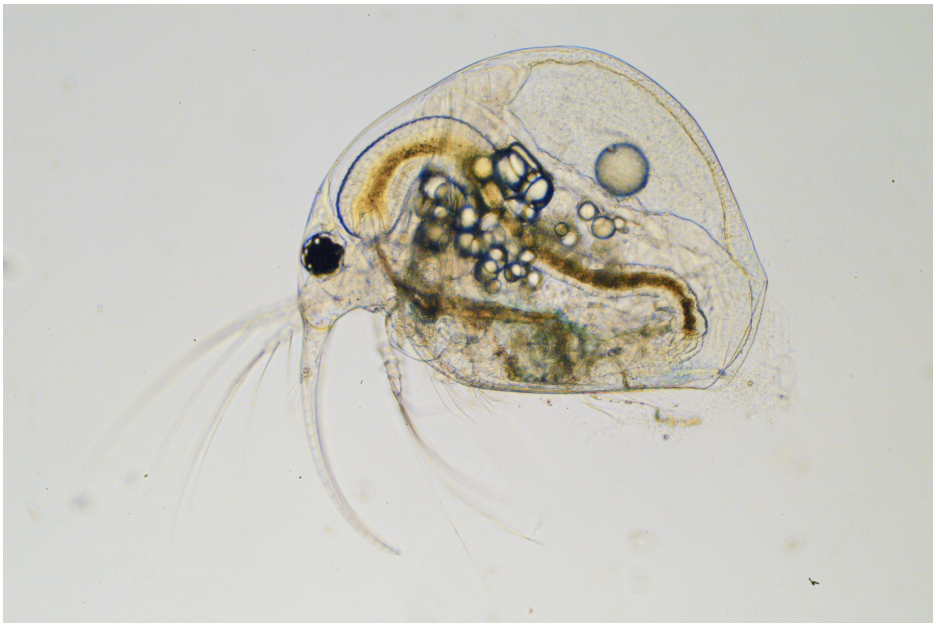


Figura 1: Ejemplo de zooplancton [1].

Antes de poder trabajar con el conjunto de imágenes final, se hace uso de Flowcam [12]. Este sistema de análisis de partículas por imágenes permite la identificación y clasificación no solo de zooplancton, sino de fitoplancton también. Es en el observatorio belga LifeWatch en donde las muestras utilizadas en este trabajo son tomadas usando nueve estaciones situadas en

la parte belga del mar del Norte. Las muestras son pasadas a través del sistema Flowcam, siendo fotografiada cada una de sus partículas. Posteriormente dichas partículas son identificadas de forma semiautomática en base a parámetros morfológicos, como la simetría, y a su fluorescencia utilizando un software específico de Flowcam. De esta manera, las imágenes utilizadas en este trabajo han sido previamente tratadas con el procedimiento comentado, para posteriormente ser posible la aplicación de técnicas de *deep learning*.

2. Tecnologías

2.1. Lenguajes de programación

2.1.1. Python

Python [13] es un lenguaje de programación de alto nivel y de propósito general, cuya idea principal es permitir a través de su sintaxis expresar conceptos de manera sencilla en una pocas líneas de código. Soporta múltiples paradigmas de programación, como por ejemplo, programación orientada a objetos, programación imperativa y funcional. Por lo general, y al igual que otros lenguajes de programación dinámicos, Python es usado de manera frecuente como un lenguaje para desarrollar *scripts* que permitan automatizar tareas concretas. De cara al uso en el campo del *machine learning* y el *deep learning*, cabe destacar la cantidad de paquetes existentes como: scikit-learn [14], Tensorflow [15], Mxnet [16], Theano [17], Lasagne [18]. Siendo estos dos últimos usados para la realización del trabajo, Theano como framework principal para el entrenamiento de las redes neuronales, y Lasagne como librería que corre por encima de Theano facilitando la construcción y manejo de las propias redes neuronales.

2.2. Sistemas de bases de datos

En cuanto a la infraestructura que da soporte al almacenamiento de todas las imágenes utilizadas durante el trabajo, se hace uso de una base de datos no relacional proporcionada de forma remota por el IFCA (Instituto de Física de Cantabria).

2.2.1. MongoDB

MongoDB [19] es un sistema de bases de datos no relacional (NoSQL) orientado principalmente al almacenamiento de documentos en un formato similar a JSON (JavaScript Object Notation) denominado BSON (Binary JSON). Mientras las bases de datos relaciones poseen tablas, MongoDB tiene colecciones de documentos en las que se utiliza la combinación clave-valor para almacenar datos y acceder a ellos.

2.2.2. GridFS

GridFS [20] es un protocolo para almacenar grandes archivos que se ejecuta por encima de MongoDB, para ello divide cada uno de los ficheros en una serie de fragmentos y los almacena como si se tratase de ficheros independientes. Además, se guardan metadatos relacionados con el archivo original.

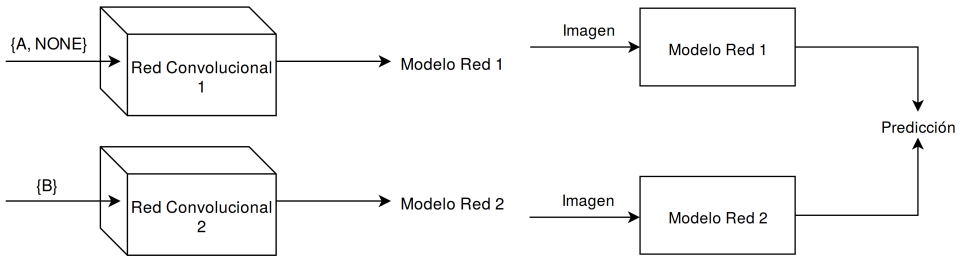
3. Metodología

Si hablamos de reconocimiento, clasificación de imágenes y visión por computador, en la actualidad podemos encontrarnos que el *estado del arte* en este campo se sitúa entorno al uso del *deep learning* [21], y más concretamente a la utilizacion de las redes convolucionales [22, 23] y las distintas variaciones que surgen de ellas. Estas son capaces de actuar de forma eficaz en multitud de campos, como por ejemplo, la imagen médica [24] o en el uso de vehículos autónomos [25] entre otros.

El presente trabajo hace uso de una de estas variantes denominada *ResNet-50* [5], descrita en mayor profundidad en la sección 3.4.1. Para ello se reutiliza código ya desarrollado para la clasificación de imágenes de plantas [26], introduciendo alguna modificación al ser pensado en principio para clasificar imágenes de plantas en color, mientras que las imágenes de zooplancton se encuentran en blanco y negro. También se hace uso de dos técnicas usadas frecuentemente en problemas de este tipo. La primera de ellas es *data augmentation*, que proporciona una serie de transformaciones a las imágenes originales para obtener una variedad más amplia sobre las mismas. Mientras que la segunda se denomina *fine-tuning*, y tiene por objetivo reutilizar los pesos entrenados para otro problema que pueda compartir características similares con el actual. Ambas técnicas son explicadas con más detalle en las secciones 3.5.1 y 3.5.2 respectivamente.

3.1. Planteamiento

Se propone dividir el problema en dos etapas. En la primera, una red clasificará entre las clases no pertenecientes al zooplancton, y en la segunda otra red ajustará la predicción entre las clases de zooplancton. La división del problema viene dado por el desbalanceado de las clases del conjunto de datos, en el que por lo general la mayor parte del conjunto de datos está compuesto por imágenes no pertenecientes al zooplancton, como pueden ser residuos orgánicos o fibras, esto queda explicado en detalle en la sección 3.3.1. El proceso a seguir tanto para la fase de entrenamiento como para la fase de test se puede apreciar en la Figura 2a, en donde A es el conjunto de clases mayoritarias/no zooplancton y B el de minoritarias/zooplancton, mientras que NONE equivale a una clase extra introducida en la primera red para indicar que la clase no pertenece al conjunto de clases que no son de zooplancton.



(a) Fase de entrenamiento (b) Fase de test

Figura 2: Metodología planteada para el entrenamiento y test. Siendo (a) la fase perteneciente al entrenamiento y (b) la fase perteneciente al test.

Con esto se pretende realizar un filtrado a través de la primera red que contiene las imágenes no pertenecientes a especies de zooplancton, dejando la segunda red para un entrenamiento más concreto sobre las imágenes que corresponden a categorías de zooplancton. De esta manera, también es posible minimizar el desbalanceado de las clases comentado anteriormente, ya que la mayor parte no se corresponden con imágenes de zooplancton, tal y como se describe en la sección 3.3.

Por otro lado, la Figura 2b muestra el planteamiento propuesto para la parte de test, en donde las imágenes son pasadas a través de ambos modelos generados anteriormente, obteniendo la predicción final de la combinación de la predicción local de ambos.

3.2. Extracción de imágenes

Uno de los pasos iniciales es la extracción de todas las imágenes contenidas en la base de datos no relacional a subcarpetas para cada una de las categorías, como queda representado en la Figura 3. Al tratarse de una cantidad importante de imágenes, en concreto 850306, es un proceso relativamente lento y propenso a posibles errores durante la realización del mismo. Debido a este motivo se implementa un sistema que vaya almacenando en un fichero los MD5 de cada una de las imágenes para en caso de ocurrir algún problema durante la extracción poder restablecer el proceso en el lugar donde se produjo el incidente, filtrando en la consulta a la base de datos las imágenes ya extraídas con anterioridad. Además, se permite cancelar de forma intencionada el proceso en cualquier momento utilizando el sistema anteriormente descrito, y retomarlo en el punto donde se dejó.

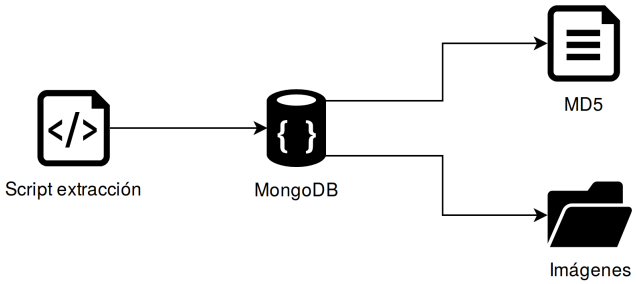


Figura 3: Flujo de extracción de las imágenes procedentes de la base de datos no relacional.

3.3. Conjunto de datos

Los datos utilizados para la realización del presente trabajo son 850306 imágenes de 25 clases distintas, 4 de ellas no pertenecientes al zooplancton y las 21 restantes que sí pertenecen a especies de zooplancton, cada una de las cuales posee la etiqueta correspondiente a su categoría. En este caso las 4 categorías que no se corresponden con algún tipo de especie de zooplancton son: *Detritus*, *Artefacts*, *Fibres* y *Others*. Estas categorías forman parte de residuos orgánicos, objetos desconocidos o fibras.

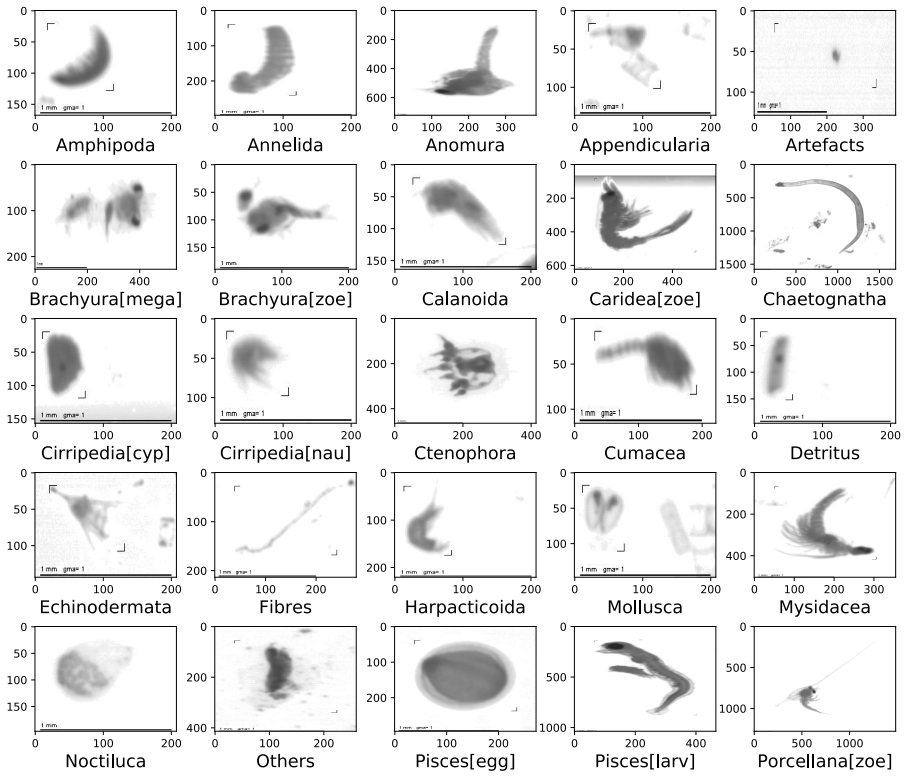


Figura 4: Imágenes de ejemplo de cada una de las clases.

En la Figura 4 se puede obtener una vista previa de cada una de las categorías presentes en el conjunto de datos, para ello se muestra un ejemplo de cada clase. También se puede apreciar como las imágenes poseen diferentes tamaños, desde la mayor de ellas con un tamaño de 15882x21176, hasta la menor con un tamaño de 114x38, aunque el tamaño de imagen más repetido a lo largo del conjunto de datos es 208x129. A pesar de los diferentes tamaños de las imágenes, estas son homogeneizadas a un tamaño de 224x224 antes de ser utilizadas para la fase del entrenamiento. En cuanto a la escala de colores se refiere, todas se encuentran representadas a través de la escala de grises.

3.3.1. Distribuciones

Al tratarse de un conjunto de datos relativamente amplio, es fácil encontrar categorías desbalanceadas en cuanto a número de imágenes, tal y como se muestra en la Figura 5, en donde se aprecia como la clase *Detritus* con 441675 imágenes acapara un 51,94 % del conjunto total de datos, haciendo evidente que se trata de un conjunto de imágenes totalmente desbalanceado, este hecho incrementa la complejidad del problema al tener que tratar

el desbalanceado de las clases.

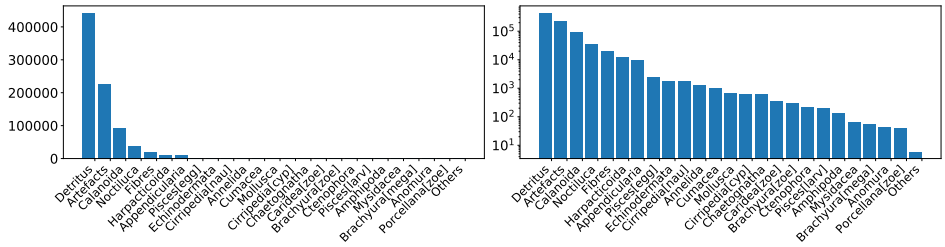


Figura 5: Distribución global.

Debido a disponer de un tiempo y unos recursos computacionales limitados, el uso de cualquier tipo de validación cruzada queda descartado, pasando a dividir el conjunto de datos en entrenamiento, validación y test. Para mantener las distribuciones originales, se utiliza muestreo estratificado para los tres conjuntos. Esta técnica permite que cada una de las categorías tenga la representación apropiada en base a la distribución original. En el caso del conjunto de entrenamiento, el tamaño corresponde al 80 % de la muestra original tal y como se puede ver en la Figura 6, con un tamaño total de 680242 imágenes.

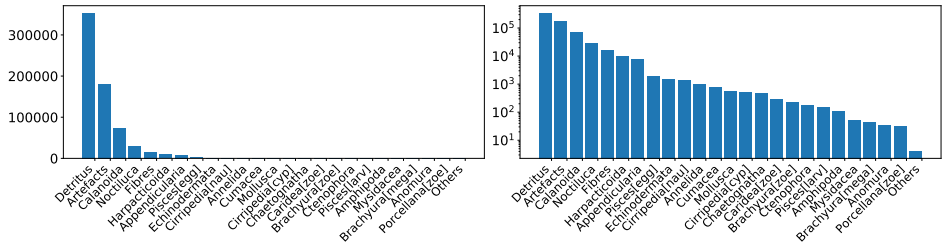


Figura 6: Distribución del entrenamiento.

En cuanto a los conjuntos de validación y de test, se toma el 10 % para cada uno de ellos, resultando en las distribuciones mostradas en las Figuras 7 y 8 respectivamente, cada una de las cuales posee 85302 imágenes en su totalidad.

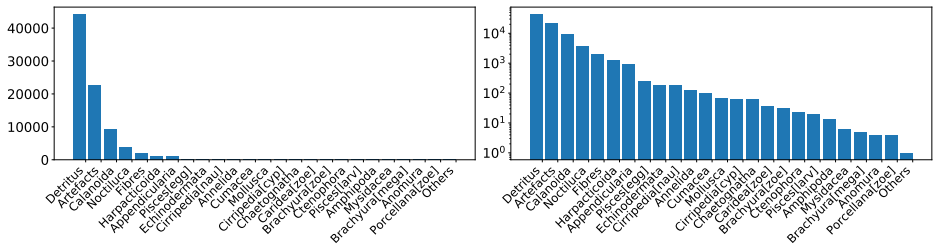


Figura 7: Distribución de la validación.

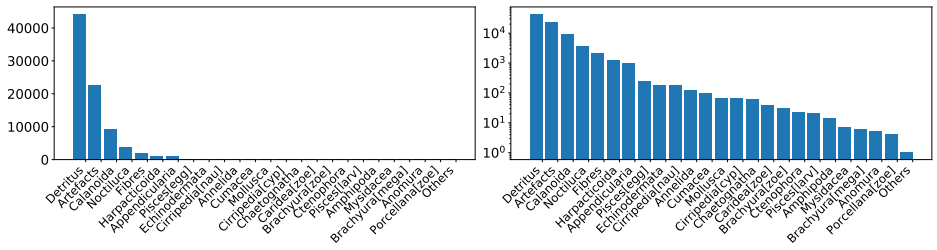


Figura 8: Distribución del test.

El número de imágenes de cada especie, junto con sus respectivas distribuciones en los conjuntos de entrenamiento, validación y test, y con el porcentaje de cada una de ellas queda recogido de forma general en la Tabla 1. En ella se aprecia el desbalanceado existente entre las clases, haciendo que entre las 7 clases con más imágenes sumen un 98.6 % aproximadamente del total, dejando el 1.4 % restante para las otras 18 clases.

	Imágenes			Porcentaje		
Clase	Entr.	Val.	Test.	Entr.	Val.	Test.
Detritus	353340	44168	44167	51,9	51,9	51,9
Artefacts	180718	22590	22590	26,6	26,6	26,6
Calanoida	73816	9227	9227	10,9	10,9	10,9
Noctiluca	29376	3672	3672	4,3	4,3	4,3
Fibres	16269	2034	2033	2,4	2,4	2,4
Harpacticoida	9693	1212	1211	1,4	1,4	1,4
Appendicularia	7672	959	959	1,1	1,1	1,1
Pisces[egg]	1966	246	246	0,3	0,3	0,3
Echinodermata	1473	184	184	0,2	0,2	0,2
Cirripedia[nau]	1432	179	179	0,2	0,2	0,2
Annelida	1006	126	126	0,1	0,1	0,1
Cumacea	790	99	99	0,1	0,1	0,1
Mollusca	549	69	68	0,08	0,08	0,08
Cirripedia[cyp]	506	63	64	0,07	0,07	0,08
Chaetognatha	489	61	61	0,07	0,07	0,07
Caridea[zoe]	295	37	37	0,04	0,04	0,04
Brachyura[zoe]	238	30	30	0,03	0,04	0,04
Ctenophora	181	23	22	0,03	0,03	0,03
Pisces[larv]	159	20	20	0,02	0,02	0,02
Amphipoda	107	13	14	0,02	0,02	0,02
Mysidacea	51	6	7	0,007	0,007	0,008
Brachyura[mega]	43	5	6	0,006	0,006	0,007
Anomura	36	4	5	0,005	0,005	0,006
Porcellana[zoe]	33	4	4	0,005	0,005	0,005
Others	4	1	1	0,001	0,001	0,001
Total	680242	85032	85032			

Tabla 1: Número de imágenes de las distribuciones de cada categoría junto con sus correspondientes porcentajes.

3.4. Redes convolucionales

Las redes convolucionales surgen como un subtipo de las redes neuronales artificiales tradicionales con uso extendido en el campo de la visión artificial. Una de las primeras arquitecturas más destacadas fue LeNet-5 [2], representada en la Figura 9, seguida de otras en los últimos años como AlexNet [27], VGG-16 [28] o ResNet [5]. Sobre esta última se centra en mayor detalle la sección 3.4.1. En la actualidad, la arquitectura Inception-v4 [29] junto con alguna variante de la ResNet, como es ResNeXt [30], se encuentran entre las más usadas para tareas de clasificación de imágenes.

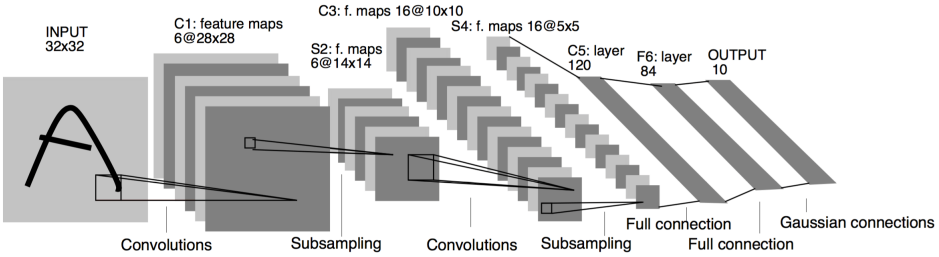


Figura 9: Red convolucional LeNet-5 [2].

La principal diferencia con las redes neuronales tradicionales reside en la incorporación de la operación denominada convolución, en la que se multiplica elemento a elemento la imagen resultante hasta el momento con una serie de matrices de pesos de tamaño f x f conocidos como filtros o *kernels*.

$$\begin{pmatrix} 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} * \begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 4 & 3 & 4 & 1 \\ 1 & 2 & 4 & 3 & 3 \\ 1 & 2 & 3 & 4 & 1 \\ 1 & 3 & 3 & 1 & 1 \\ 3 & 3 & 1 & 1 & 0 \end{pmatrix}$$

7 x 7 3 x 3 5 x 5

Figura 10: Ejemplo de convolución [3] utilizando una representación de 7×7 con un filtro de 3×3, dando resultado a una representación de 5×5.

De forma opcional, se pueden utilizar dos parámetros habituales en las operaciones de convolución. El primero de ellos denominado *padding* incrementa el número de píxeles de cada uno de los bordes de la representación, por lo general estableciendo el valor de estos a 0 (*zero padding*). Con esto se consigue que los píxeles cercanos a los bordes no se vean infrarrepresentados con respecto a los píxeles que ocupan posiciones más centrales, y permite que el paso a través de múltiples capas no reduzca en exceso el tamaño de la representación final. El otro parámetro se denomina *stride*, su valor establece el número de píxeles a desplazarse por el filtro, tanto de forma horizontal como vertical. Por lo tanto, las dimensiones resultantes se pueden calcular como,

$$\left[\frac{n+2p-f}{s}+1\right] \times \left[\frac{n+2p-f}{s}+1\right]$$

siendo n la dimensión de la entrada, f la dimensión del filtro, p el valor del *padding* y s el del *stride*.

La mayor parte de las arquitecturas de redes convolucionales incorporan una operación denominada *pooling*, la cual permite reducir la dimensionalidad de la representación existente. Al igual que con la operación de convolución, se define un tamaño de filtro que opera tomando el máximo (*max pooling*), o el promedio (*average pooling*) de la región sobre la que se aplica la operación, como se muestra en la Figura 11. A diferencia de la convolución, el *pooling* no requiere de un aprendizaje de pesos o parámetros de la red, por lo que su complejidad computacional es prácticamente nula.

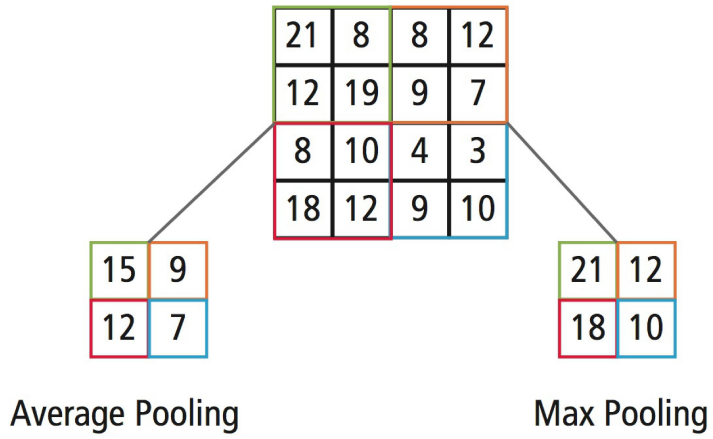


Figura 11: Dos formas distintas de hacer la operación de *pooling* [4], utilizando *average pooling* o *max pooling*.

Aplicada la convolución y el *pooling* sobre cada una de las capas, se utiliza una función de activación, por lo general ReLU (*Rectified Linear Unit*) [31], que permite evitar el problema del desvanecimiento de gradiente (*vanishing gradient problem*) [32] producido cuando el número de capas es elevado y se utilizan funciones como la sigmoide o la tangente hiperbólica. Los valores resultantes de aplicar esta serie de operaciones se convierten a un vector que se pasa como entrada a una red densamente conectada, la cual puede tener múltiples capas ocultas y una salida que utiliza *softmax* en caso de tratarse de un problema con más de dos clases al proporcionar una distribución de probabilidad sobre las clases utilizadas, o simplemente una función de activación sigmoide en caso de tratarse de un problema de clasificación binaria.

3.4.1. Redes residuales

Una de las variantes más utilizadas hoy en día sobre las redes convolucionales es el uso de bloques residuales que da lugar a las denominadas redes residuales [5], o más comúnmente conocidas como ResNet, a las que acompaña un número indicando la cantidad de bloques residuales utilizados.

En el presente trabajo se usan 50 de estos bloques, pasando a denominarse la red como ResNet-50.

Este tipo de variante permite entrenar redes con una mayor profundidad, ya que su objetivo principal es evitar la degradación que se produce al tener una red con un número relativamente alto de capas ocultas. Este problema no viene producido por sobreajuste u *overfitting* [33], sino que el añadir más capas hace que incluso el error de entrenamiento se vea incrementado, como queda reflejado en la Figura 12, perteneciente a uno de los experimentos llevados a acabo en el artículo original sobre las redes residuales.

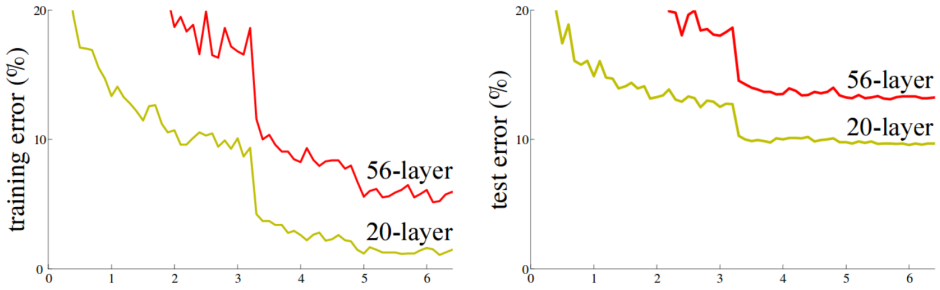


Figura 12: Error en función del número de capas para una red no residual [5].

La idea principal reside en tratar el problema de la degradación comentado anteriormente. Para ello se define $H(x)$ como la función que representaría el aprendizaje por parte de la red con sus correspondientes capas no lineales, siendo x la entrada de la primera de ellas. Dicha función viene dada por $H(x) = F(x) + x$, partiendo de la hipótesis de que es más fácil optimizar la parte residual que la función original de aprendizaje de la red, por lo que el residuo se obtiene con $F(x) = H(x) - x$.

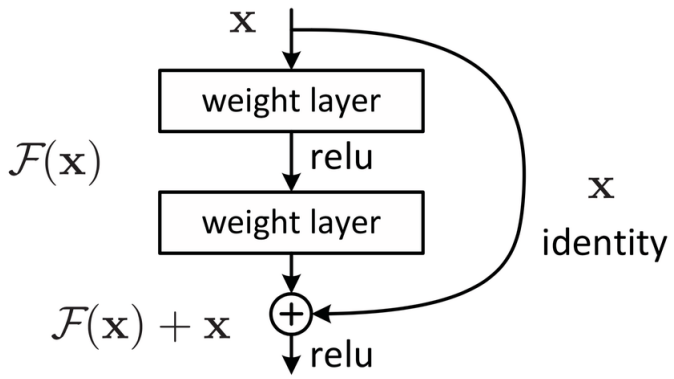


Figura 13: Bloque residual [5].

Esta formulación permite establecer una conexión directa entre conjuntos de capas, siendo posible añadir la función identidad de un conjunto de capas previas a otro conjunto de capas posteriores como se muestra en la Figura 13, con un bloque residual en el que $F = W_2\sigma(W_1x)$ con σ siendo la función de activación, en este caso ReLU [31]. Por lo tanto, cada uno de los bloques residuales viene dado por,

$$y = F(x, \{W_i\}) + x$$

en donde $F(x, \{W_i\})$ representa el mapeo residual a aprender por la red.

En caso de no coincidir las dimensiones de x y F se introduce una matriz W_s, quedando la formulación final de cada bloque residual como,

$$y = F(x, \{W_i\}) + W_sx$$

3.5. Entrenamiento

Para la fase del entrenamiento, se hace uso de pesos preentrenados sobre el conjunto de datos ImageNet [34], esto proporciona los beneficios del *fine-tuning* comentados en la sección 3.5.2. En la Figura 2a se puede visualizar el planteamiento para la parte del entrenamiento, ya introducido de forma breve en la sección 3.1. En dicha figura se aprecia como la primera red entrena con las imágenes de las clases que no son de zooplancton, denotadas como A, junto con la clase NONE. La otra de las redes entrena con las imágenes de las clases de zooplancton, denotadas como B. Es importante resaltar que A contiene un conjunto de clases, mientras que NONE es una única clase, quedando denotado de esta manera para simplificar la Figura 2a, mientras que lo mismo ocurre para la segunda red, en la que el conjunto de clases viene dado por B.

A través de las Tablas 2 y 3 se pueden visualizar las clases utilizadas para cada una de las redes respectivamente, junto con el número de imágenes y el porcentaje que representa cada una de ellas. La primera abarca un total de 680242 imágenes para el entrenamiento, mientras que la segunda cuenta con 129915, correspondientes a las imágenes de la clase NONE en la primera red. Es importante destacar que la clase *Others*, que no pertenece al zooplancton, se ha incluido en las clases de zooplancton de la segunda red, debido a que en el conjunto de imágenes original apenas cuenta con 6 imágenes.

Red 1	**Imágenes**			**Porcentaje**		
Clase	Entr.	Val.	Test	Entr.	Val.	Test
Detritus	353340	44168	44167	51,9	51,9	51,9
Artefacts	180718	22590	22590	26,6	26,6	26,6
NONE	129915	16240	16242	19,1	19,1	19,1
Fibres	16269	2034	2033	2,4	2,4	2,4
Total	680242	85032	85032			

Tabla 2: Imágenes de las categorías utilizadas para la primera red.

Red 2	**Imágenes**			**Porcentaje**		
Clase	Entr.	Val.	Test	Entr.	Val.	Test
Calanoida	73816	9227	9227	56,8	56,8	56,8
Noctiluca	29376	3672	3672	22,6	22,6	22,6
Harpacticoida	9693	1212	1211	7,5	7,5	7,5
Appendicularia	7672	959	959	5,9	5,9	5,9
Pisces[egg]	1966	246	246	1,5	1,5	1,5
Echinodermata	1473	184	184	1,1	1,1	1,1
Cirripedia[nau]	1432	179	179	1,1	1,1	1,1
Annelida	1006	126	126	0,8	0,8	0,8
Cumacea	790	99	99	0,6	0,6	0,6
Mollusca	549	69	68	0,4	0,4	0,4
Cirripedia[cyp]	506	63	64	0,4	0,4	0,4
Chaetognatha	489	61	61	0,4	0,4	0,4
Caridea[zoe]	295	37	37	0,2	0,2	0,2
Brachyura[zoe]	238	30	30	0,2	0,2	0,2
Ctenophora	181	23	22	0,1	0,1	0,1
Pisces[larv]	159	20	20	0,1	0,1	0,1
Amphipoda	107	13	14	0,1	0,1	0,1
Mysidacea	51	6	7	0,04	0,04	0,04
Brachyura[mega]	43	5	6	0,03	0,03	0,04
Anomura	36	4	5	0,03	0,02	0,03
Porcellana[zoe]	33	4	4	0,03	0,02	0,02
Others	4	1	1	0,003	0,006	0,006
Total	129915	16240	16242			

Tabla 3: Imágenes de las categorías utilizadas para la segunda red.

Para entrenar cada una de las redes es necesario establecer el número de épocas y el tamaño de *batch* que se utiliza. El número de épocas indica el número de veces que el conjunto completo de los datos es pasado a través de la red, mientras que el tamaño de *batch* determina el número de imágenes que

se utiliza para calcular los gradientes necesarios para actualizar los pesos de la red, haciendo que múltiples *batches* compongan una única época, pasando a denominarse *mini-batches*. El uso de *mini-batches* es necesario al tratarse de un problema con un número relativamente alto de imágenes, por lo que no establecer un tamaño de *batch* significaría tener que cargar todas las imágenes a la vez en memoria, resultando computacionalmente muy costoso para el problema planteado.

En este caso se ha decidido utilizar un tamaño de *batch* de 64 para las dos redes, haciendo que únicamente se cargue en memoria ese número de imágenes al mismo tiempo y permitiendo actualizar los pesos de la red una vez se han calculado los gradientes necesarios de cada *mini-batch*. Por otra parte, el número de épocas para la primera red se ha establecido en 20, mientras que, en el caso de la segunda, al poseer un número inferior de imágenes, es posible en cuanto a recursos computacionales se refiere, entrenarla durante un número mayor de épocas, siendo 150 el valor elegido. Ambos valores han sido seleccionados teniendo en cuenta los recursos computaciones disponibles y teniendo por objetivo realizar la fase de entrenamiento en un tiempo que no superase la semana de ejecución.

Con el objetivo de reducir el impacto negativo que supone el trabajar con un conjunto de datos tan desbalanceado, a la función de coste por la que se rige todo el proceso de optimización, en este caso *Categorical Cross Entropy* [35], se le incorpora una ponderación sobre las clases o también denominado como *class weights*. Esto permite que las muestras de las categorías con un menor número de instancias tengan un mayor impacto sobre la función de coste, mientras que las categorías predominantes posean un menor peso, haciendo que el aprendizaje se produzca de una manera más balanceada.

Además, el algoritmo de optimización que ajusta los pesos de la red es Adam (*Adaptative Moment Estimation)* [36], el cual combina los beneficios de otros algoritmos y técnicas de optimización [37], como son Momentum [38] y RMSprop [39].

Para que todo el proceso de entrenamiento sea posible llevarlo a cabo en un tiempo razonable, se hace uso de la GPU Titan X Pascal, que cuenta con 12 GB de memoria RAM.

3.5.1. *Data augmentation*

A través de la técnica denominada *data augmentation* [40] es posible proporcionar a la red las imágenes originales con ciertas modificaciones. Estas modificaciones pueden ir desde rotar la imagen una serie de determinados grados, modificar los valores RGB, o aplicar un efecto de espejo o *mirroring*

sobre la imagen, como se muestra en el ejemplo de la Figura 14. Con esto se pretende conseguir mayor diversidad en el conjunto de datos a la hora de entrenar la red. También es utilizado en problemas en los que alguna clase del conjunto de imágenes es reducido, como es el caso del problema planteado en el presente trabajo.

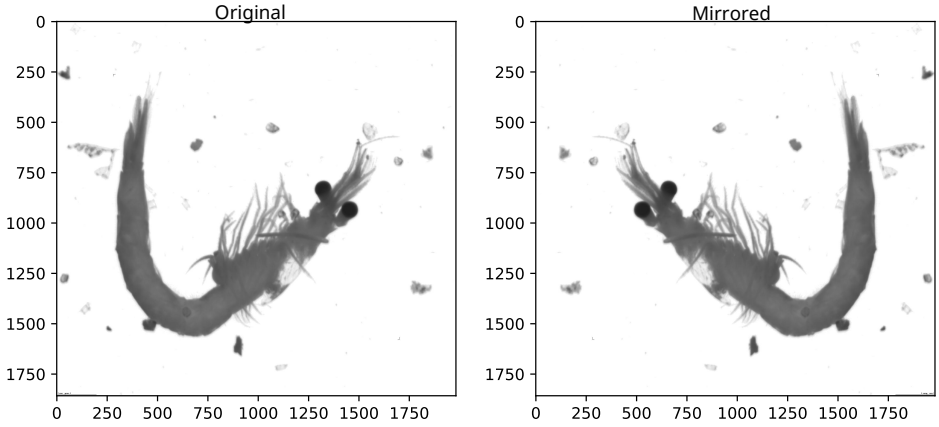


Figura 14: Ejemplo de *mirroring* sobre una de las especies.

3.5.2. *Fine-tuning*

Otra de las técnicas usadas es el *fine-tuning*, que permite utilizar una red previamente entrenada sobre con conjunto de datos relativamente amplio para otro problema, pero cuyas características básicas puedan ser comunes al problema actual y por lo tanto útiles. Para el problema tratado, el conjunto de datos sobre los que se entrenaron los pesos originales es ImageNet [34], un conjunto de datos con millones de imágenes y miles de categorías. De esta forma es más sencillo encontrar características comunes en las primeras capas de la red, por lo que son las últimas capas las que se entrenan sobre los datos del problema a tratar, ya que en ellas se pueden encontrar representaciones más abstractas, y por lo tanto, más específicas del problema.

3.6. Test

Generados los modelos que permiten la clasificación de las especies, la forma de clasificar nuevas imágenes no vistas ni en el conjunto de entrenamiento ni en el de validación es a través de la metodología planteada en la Figura 2b. En ella se pasa la nueva imagen al modelo de la primera red, el cual trata de clasificar entre las clases que no son de zooplancton o NONE. Posteriormente la misma imagen es pasada a la segunda red que se encarga de clasificar entre las especies de zooplancton. Es imprescindible pasar la imagen a la segunda red para poder calcular el Top-5, explicado en detalle

en la sección 4, al únicamente tener 4 clases en la primera red contando la clase NONE (que no contabiliza para el Top-5).

Si en vez de calcular hasta el Top-5, se calculase únicamente el Top-1, no sería necesario pasar las imágenes en las que la predicción no fuese NONE a la segunda red, como se muestra en la Figura 15, ahorrando tiempo de cómputo.

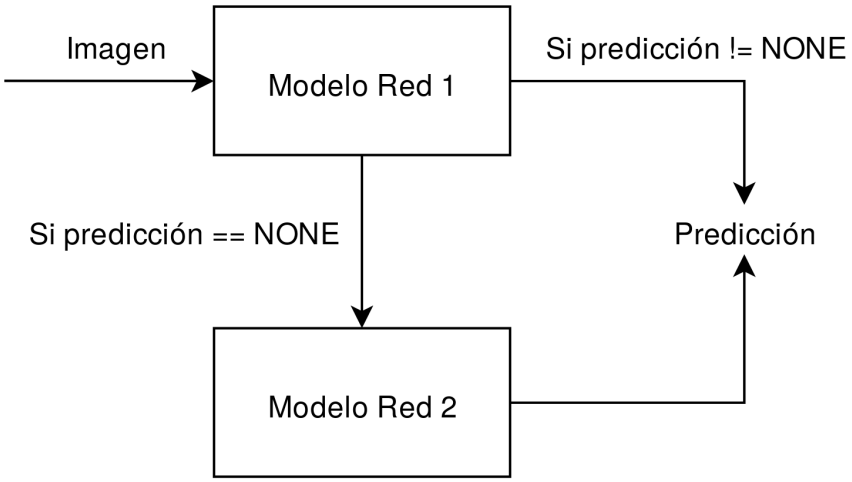


Figura 15: Metodología en caso de testear con el Top-1, en donde solo es necesario usar la segunda red si la clase con mayor probabilidad es NONE.

A modo de ejemplo, la Figura 16 recoge parte del proceso de predicción final a seguir para una imagen. En ella se encuentran, de forma ordenada de izquierda a derecha, tanto el vector de predicciones de las clases como el vector que contiene la probabilidad de cada una de ellas, mientras que los valores de la clase NONE se encuentran resaltados.

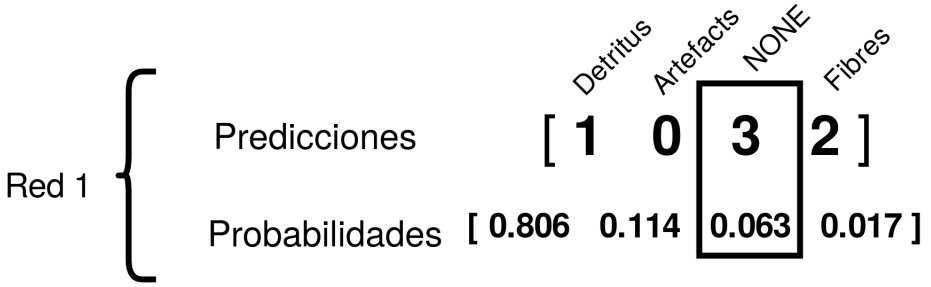


Figura 16: Ejemplo de predicción para la primera red, en la que se obtiene un vector de clases y otro de probabilidades.

Lo mismo ocurre para las predicciones de la segunda red, ilustradas a

través de la Figura 17. Con la diferencia de que los índices tienen que ser normalizados para conseguir la sensación de una única red global, esto se consigue sumando el número de clases de la primera red (sin contar NONE) a cada índice. Además, las probabilidades de la clase NONE se reparten entre las clases de la segunda red, para ello se multiplica cada probabilidad obtenida en esta nueva red por la probabilidad de la clase NONE obtenida en la primera.

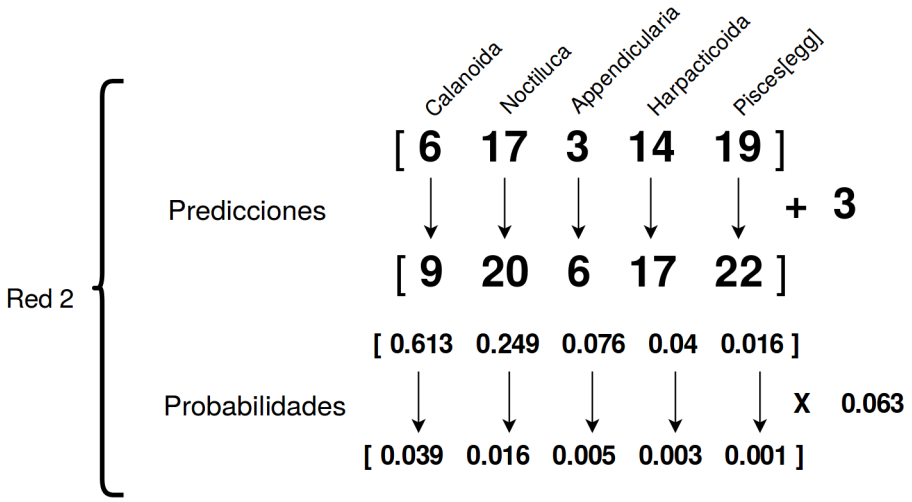


Figura 17: Ejemplo de predicción para la segunda red, en la que se obtiene un vector de clases y otro de probabilidades.

Por último, es necesario ordenar las nuevas probabilidades obtenidas junto con parte de las ya existentes de la primera red para generar la predicción final para la imagen correspondiente, tal y como se muestra en la Figura 18. Este tipo de implementación permite calcular de forma sencilla la métrica Top-K, descrita en la sección 4.

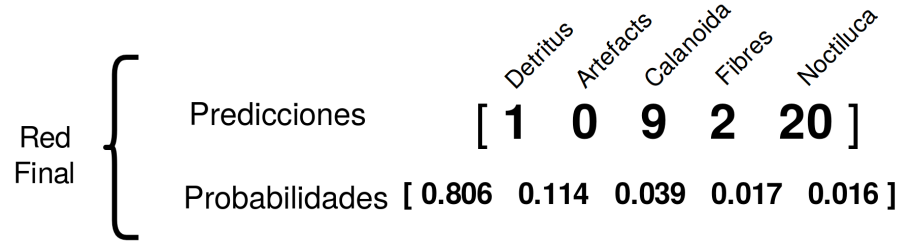


Figura 18: Ejemplo de predicción final, en la que se combinan las predicciones de las dos redes.

4. Resultados

La forma de obtener una idea del rendimiento que puedan proporcionar los modelos sobre conjuntos de datos independientes en la fase de entrenamiento, es a través de la utilización de un conjunto de datos de validación sobre el que evaluar los modelos. En la Figura 19 se recoge tanto la pérdida que tiene la primera de las redes a lo largo de las épocas como el acierto obtenido, con la única diferencia de que para el entrenamiento se grafica el resultado de cada *mini-batch* y para la validación los valores graficados son la media de cada época. En ella se puede ver como no se produce sobreajuste sobre el conjunto de validación, sino que la pérdida y acierto de ambos conjuntos de datos obtienen valores similares, haciendo pensar que existe todavía un margen de mejora posible si el entrenamiento se prolongase a lo largo de un número mayor de épocas.

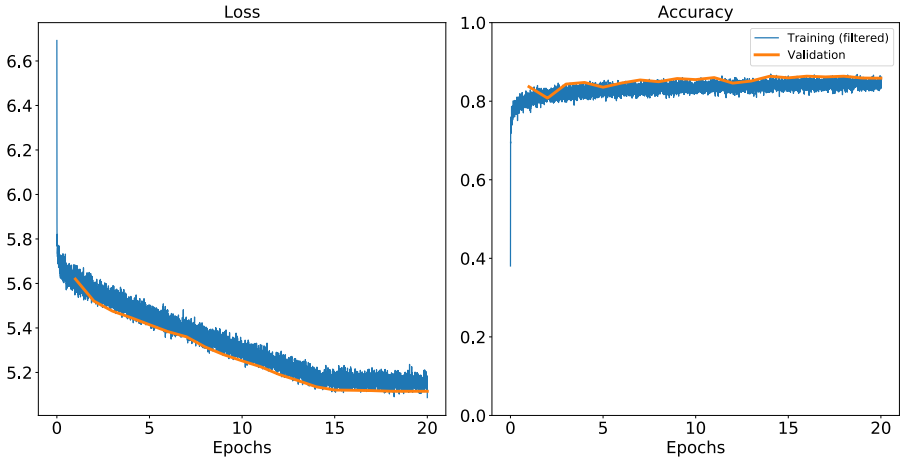


Figura 19: Pérdida y acierto de la primera red, tanto para el entrenamiento como para la validación.

Para la segunda red, la evaluación del proceso de entrenamiento se puede seguir a través de la Figura 20. En este caso la pérdida de la validación y el entrenamiento se va distanciando a medida que transcurren las épocas, esto podría ser un signo de que podría sobreajustar si el entrenamiento se hubiese realizado durante un número mayor de épocas. Para el acierto, no es hasta entorno a la época 110 que la red parece estabilizarse en su rendimiento sobre el conjunto de validación, ya que hasta ese momento el comportamiento estaba siendo muy errático.

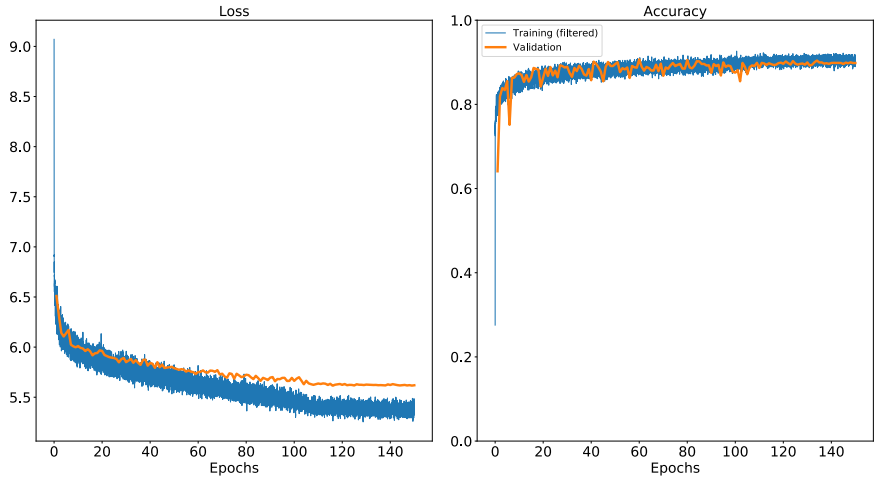


Figura 20: Pérdida y acierto de la segunda red, tanto para el entrenamiento como para la validación.

Entrenados ambos modelos y evaluados sobre sus respectivos conjuntos de validación, se pasa el conjunto de imágenes de test a través de ellos. En este caso se toma como métrica de que tan bien ha sido su rendimiento haciendo uso del Top-K, siendo $K = \{1, 2, 3, 4, 5\}$. Esta métrica permite obtener el acierto teniendo el cuenta las K clases con mayor probabilidad para cada una de las imágenes del conjunto de test, por lo que si la clase verdadera a la que pertenece se encuentra dentro de las K con más probabilidades se cuenta como acierto. Se ha tomado 5 como el mayor número utilizado para el Top, al tratarse de un problema con pocas categorías, por lo que la utilización de un Top más amplio no fuese relevante al situarse el acierto cercano al 100 % con el Top-5, tal y como se muestra en la Tabla 4. En ella incluso se puede apreciar como utilizando el Top-3 ya es suficiente para alcanzar un rendimiento, en cuanto a acierto, superior al 99 %.

	Acierto
Top-1	85.79 %
Top-2	96.79 %
Top-3	99.05 %
Top-4	99.59 %
Top-5	99.77 %

Tabla 4: Acierto desde el Top-1 hasta el Top-5 para el conjunto de test.

El acierto a nivel individual de cada una de las categorías es visible a través de la Tabla 5. En ella queda reflejado la diferencia de acierto a lo largo de las diferentes categorías.

Clase	Acierto
Artefacts	90.02 %
Fibres	89.08 %
Calanoida	88.24 %
Cumacea	85.86 %
Harpacticoida	85.30 %
Detritus	84.71 %
Noctiluca	79.30 %
Amphipoda	78.57 %
Appendicularia	77.79 %
Caridea[zoe]	72.97 %
Mysidacea	71.43 %
Brachyura[zoe]	70.00 %
Pisces[larv]	60.00 %
Echinodermata	57.61 %
Chaetognatha	55.74 %
Ctenophora	54.55 %
Cirripedia[nau]	49.16 %
Cirripedia[cyp]	48.44 %
Annelida	45.24 %
Anomura	40.00 %
Brachyura[mega]	33.33 %
Porcellana[zoe]	25.00 %
Pisces[egg]	22.76 %
Mollusca	10.29 %
Others	0.00 %

Tabla 5: Acierto para cada categoría sobre el conjunto de test.

Una forma de visualizar en mayor profundidad el rendimiento de las predicciones sobre cada una de las clases es a través de la matriz de confusión. En este caso se hace uso de una matriz de confusión que recoge las probabilidades de cada predicción y posteriormente las normaliza en función del número de muestras de cada una de las clases, tal y como se muestra en la Figura 21.

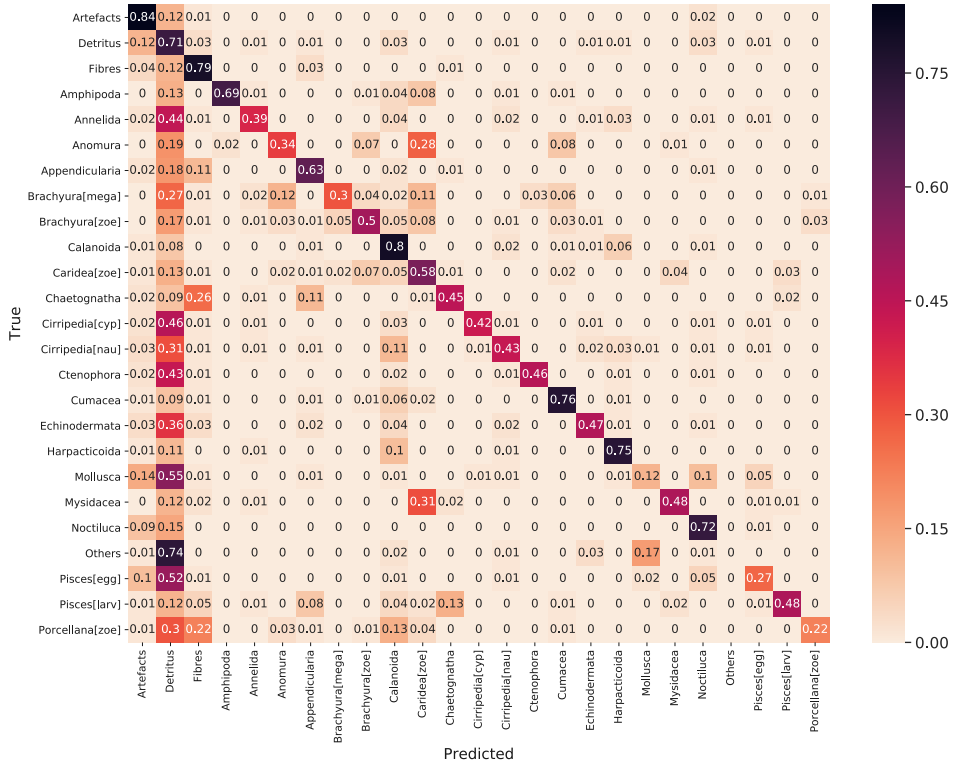


Figura 21: Matriz de confusión de probabilidades.

Es evidente que en las predicciones sigue existiendo cierta descompensación, haciendo que gran parte de las imágenes sean clasificadas como *Detritus*, como se muestra en la Figura 22, sin serlo en realidad, aunque existen ciertas excepciones como *Calanoida* o *Cumacea*.

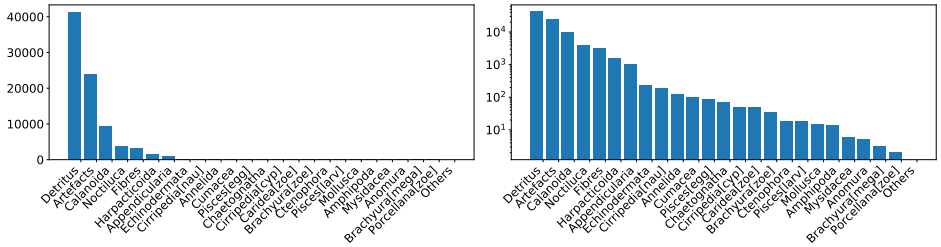


Figura 22: Distribución de las predicciones.

Para un mayor entendimiento de los resultados obtenidos, se hace uso también de otras métricas como *Precision*, *Recall* y *F1 Score*, tanto en sus versiones Macro como Micro [41]. Estas métricas vienen dadas por,

$$Macro - Precision = \frac{\sum_{i=1}^{l} \frac{tp_i}{(tp_i + fp_i)}}{l}$$

$$Macro - Recall = \frac{\sum_{i=1}^{l} \frac{tp_i}{(tp_i + fn_i)}}{l}$$

$$Micro - Precision = \frac{\sum_{i=1}^{l} tp_i}{\sum_{i=1}^{l} (tp_i + fp_i)}$$

$$Micro - Recall = \frac{\sum_{i=1}^{l} tp_i}{\sum_{i=1}^{l} (tp_i + fn_i)}$$

en donde tp_i, fp_i, fn_i se corresponden con los verdaderos positivos, falsos positivos y falsos negativos respectivamente para cada clase, mientras que l indica el número de clases. Mientras que *F1 Score*, al ser la media armónica de *Precision* y *Recall*, se calcula de la misma forma independientemente de la versión Macro o Micro,

$$F1\ Score = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall}$$

En la Tabla 6 quedan reflejados los valores obtenidos para las métricas descritas anteriormente. En la versión Macro las categorías son tratadas por igual sin tener en cuenta el número de imágenes de cada una, mientras que la versión Micro si tiene en cuenta este aspecto. Por lo tanto, los resultados de la versión Macro son inferiores en cuanto a rendimiento ya que las clases que mejor clasifican tienen tanta importancia en el resultado final como las clases que peor lo hacen. De la misma forma se puede apreciar como los valores Micro y el Top-1 son equivalentes entre sí.

Macro			**Micro**		
Precision	**Recall**	**F1 Score**	**Precision**	**Recall**	**F1 Score**
62.5 %	59.02 %	0.607	85.79 %	85.79 %	0.858

Tabla 6: Resultados utilizando las métricas Macro y Micro.

Los resultados obtenidos se pueden situar en un contexto de comparación con otros resultados y técnicas distintas para resolver el mismo problema. En este caso se propone comparar los resultados obtenidos con otros que tratan de clasificar imágenes de zooplancton [42] utilizando SVM (*Support Vector Machine*) [43], MKL (*Multiple Kernel Learning*) [44] y un sistema base de referencia (*Baseline System*) [45] basado en los atributos más relevantes de las imágenes, todo ello sobre el conjunto de 3771 imágenes ZooScan, y por lo tanto un conjunto ampliamente más reducido que el utilizado en el presente trabajo, en el que las imágenes fueron tomadas en la bahía de Villefrance Sur

Mer en Francia [46]. Las métricas utilizadas en la comparación vienen dadas por la versión Macro, que como se comentaba anteriormente no tiene en cuenta el desbalanceado de los datos, pero son las métricas requeridas para poder comparar los resultados obtenidos con los ya publicados. A través de la Tabla 7 se aprecia esta comparación, en donde es evidente que al tratarse de conjunto de imágenes que sufre de un grado alto de desbalanceado, los valores obtenidos se encuentran por debajo de los ya publicados. En ella se incluye también $1 - Precision$, considerándose como una medida de error de *Precision* que es utilizada en la publicación.

Técnica	**Recall**	**1-Precision**	**F1 Score**
ResNet-50 (este trabajo)	59.02 %	37.50 %	0.607
Sistema base de referencia [42]	80.77 %	16.28 %	0.822
SVM (Lineal, C=10) [42]	85.52 %	16.01 %	0.847
MKL(Polinomial,C=100)[42]	**86.79 %**	**11.63 %**	**0.876**

Tabla 7: Comparación de los resultados Macro entre diferentes técnicas.

Es importante aclarar que tanto este trabajo como el utilizado en la comparación comparten el mismo objetivo, clasificar imágenes de zooplancton. Sin embargo, existen grandes diferencias en cuanto a número de imágenes utilizadas y a la distribución de las mismas, haciendo que la publicación no sufra de desbalanceado en el mismo grado que sí ocurre con el presente trabajo, y tratando 20 clases distintas por las 25 aquí recogidas. Además, las métricas Macro no favorecen en absoluto a un conjunto de datos desbalanceado, haciendo que los resultados recogidos en la Tabla 7 puedan parecer pobres.

La solución a este problema viene dada por comparar los resultados siguiendo las métricas Micro ya descritas anteriormente. Ha sido necesario calcular los valores Micro de los resultados publicados, ya que solo se proporcionaban los valores Macro. Para ello se han tomado las matrices de confusión publicadas, que aunque por razones desconocidas no se corresponden con las de los resultados de los mejores métodos obtenidos en la publicación, sirven como comparación con los resultados obtenidos en el trabajo. Se ha de suponer que con las matrices de confusiones de los mejores métodos los resultados Micro de la publicación hubiesen sido ligeramente mejores, pero no existe forma de reproducirlos sin esas matrices. Dicho lo cual, la comparación con los valores Micro viene dada por la Tabla 8. En ella se puede observar como los resultados se sitúan en valores muy similares a los de la publicación, pudiendo considerarse una comparación más justa que utilizando los valores Macro recogidos en la Tabla 7.

Técnica	**Recall**	**1-Precision**	**F1 Score**
ResNet-50 (este trabajo)	85.79 %	14.21 %	0.858
Sistema base de referencia [42]	81.69 %	18.31 %	0.817
SVM (Lineal, C=10) [42]	84.73 %	15.27 %	0.847
MKL(Polinomial, C=100)[42]	**87.40 %**	**12.60 %**	**0.874**

Tabla 8: Comparación de los resultados Micro entre diferentes técnicas.

5. Conclusiones

El objetivo principal del proyecto consistía en la clasificación de imágenes de distintas especies de zooplancton, por lo que se propuso la utilización de técnicas de *deep learning* que fuesen capaces de generar modelos con los que llevar a cabo dicha tarea. Cumplidos los objetivos descritos anteriormente, se pasa a valorar los resultados obtenidos para los modelos finales generados.

Los resultados en cuanto a nivel de acierto sobre los distintos Top utilizados se podrían definir como satisfactorios, teniendo en cuenta el desbalanceado tan evidente que sufre el conjunto de imágenes original, también hay que tener en cuenta que al trabajar sobre una cantidad de imágenes cercana al millón se requiere de una serie de recursos computacionales importantes para el tratamiento de tal volumen de datos. En este caso se ha dispuesto de una GPU sin la cual hubiese sido imposible llevar a cabo todo el trabajo realizado, aunque es evidente que la utilización de múltiples GPU's ayudaría a reducir tanto el tiempo de entrenamiento como el de test.

En todas las fases por las que ha pasado la realización del proyecto han surgido diferentes problemas y dificultades. Como por el ejemplo el hecho de que la extracción de las imágenes de la base de datos requiriese de varios días y en donde en principio no se contaba con ningún sistema de recuperación de las imágenes ya extraídas, teniendo que empezar el proceso desde el inicio si ocurría alguna incidencia, haciendo que el sistema de caché explicado en la sección 3.2 fuese de gran utilidad para solventar ese contratiempo. Es evidente que otro de los problemas con los que se ha tratado es el desbalanceado tan extremo que sufría el conjunto de imágenes original, y que llevó a tomar la decisión de dividir el problema en la utilización de dos redes convolucionales distintas. Este hecho del desbalanceado provocó también problemas a la hora de comparar los resultados con los ya publicados, en los que se utilizaban métricas que se podrían considerar poco adecuadas para el conjunto de imágenes utilizado, por lo que hubo que reproducir los valores publicados y convertirlos a métricas donde la comparación fuese más justa.

En cuanto a la parte personal, poder afrontar este problema me ha supuesto el tratar de primera mano con un problema real con cientos de miles de instancias y con la dificultad añadida del desbalanceado existente entre las clases. Además, me ha permitido hacer uso técnicas utilizadas en la actualidad para resolver problemas de clasificación de imágenes que requieren de un alto coste computacional, y que son tratados haciendo uso de GPU's.

De cara a la mejora del proyecto realizado, se pueden listar los siguientes puntos clave que a mi entender podrían mejorar o facilitar la mejora de lo realizado hasta la fecha:

- Incrementar el número de épocas para la primera red, en la que es evidente que al tratarse de un número relativamente grande de imágenes con respecto a la segunda, sería recomendable poder entrenarla por un tiempo más amplio, en el que probablemente el error de validación siguiese decayendo y el acierto incrementándose ligeramente. En cambio para la segunda red se puede intuir como un aumento en el número de épocas provocaría sobreajuste, tal y como se comenta en la sección 4.
- Aplicación de técnicas de *over-sampling* [47] sobre las categorías minoritarias, y de *sub-sampling* [48] en las categorías mayoritarias. Esto tendría por objetivo equilibrar en la medida en que fuese posible el número de imágenes por categoría, tratando de minimizar el problema del desbalanceado con el que se ha lidiado a lo largo de todo el trabajo.
- Realización de pruebas con diferente número clases para la primera de las redes, tratando de encontrar la combinación de clases que maximicen el acierto global.

Referencias

[1] A. Sabai, "Lake winnebago zooplankton." `https://www.flickr.com/photos/winnebagophotography/8257838959/`, 2012. [En línea; Accedido el 09-09-2018].

[2] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[3] jakun (https://tex.stackexchange.com/users/120953/jakun), "Drawing a convolution with tikz." `https://tex.stackexchange.com/questions/437007`. [En línea; Accedido el 18-08-2018].

[4] S. Hijazi, R. Kumar, and C. Rowen, "Using convolutional neural networks for image recognition," *Cadence Design Systems Inc.: San Jose, CA, USA*, 2015.

[5] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

[6] S. John Walker, "Big data: A revolution that will transform how we live, work, and think," 2014.

[7] F. Rosenblatt, *The perceptron, a perceiving and recognizing automaton Project Para*. Cornell Aeronautical Laboratory, 1957.

[8] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning internal representations by error propagation," tech. rep., California Univ San Diego La Jolla Inst for Cognitive Science, 1985.

[9] M. Cole, P. Lindeque, E. Fileman, C. Halsband, R. Goodhead, J. Moger, and T. S. Galloway, "Microplastic ingestion by zooplankton," *Environmental science & technology*, vol. 47, no. 12, pp. 6646–6655, 2013.

[10] J.-P. W. Desforges, M. Galbraith, and P. S. Ross, "Ingestion of microplastics by zooplankton in the northeast pacific ocean," *Archives of environmental contamination and toxicology*, vol. 69, no. 3, pp. 320–330, 2015.

[11] N. Khalifa, K. A. El-Damhogy, M. R. Fishar, A. M. Nasef, and M. H. Hegab, "Using zooplankton in some environmental biotic indices to assess water quality of lake nasser, egypt," *Int J Fisheries Aquat Stud*, vol. 2, no. 4, pp. 281–289, 2015.

[12] H. Wright, H. Nelson, and B. Dalton, "Improved methodology for semi-automated identification of plankton and biovolume estimation using a

digital imaging flow cytometer (flowcam)," in *American Geophysical Union, Ocean Sciences Meeting 2016, abstract# IS34A-2321*, 2016.

[13] G. Van Rossum *et al.*, "Python programming language.," in *USENIX Annual Technical Conference*, vol. 41, p. 36, 2007.

[14] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[15] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015. Software available from tensorflow.org.

[16] T. Chen, M. Li, Y. Li, M. Lin, N. Wang, M. Wang, T. Xiao, B. Xu, C. Zhang, and Z. Zhang, "Mxnet: A flexible and efficient machine learning library for heterogeneous distributed systems," *arXiv preprint arXiv:1512.01274*, 2015.

[17] R. Al-Rfou, G. Alain, A. Almahairi, C. Angermueller, D. Bahdanau, N. Ballas, F. Bastien, J. Bayer, A. Belikov, A. Belopolsky, *et al.*, "Theano: A python framework for fast computation of mathematical expressions," *arXiv preprint*, 2016.

[18] S. Dieleman, J. Schlüter, C. Raffel, E. Olson, S. K. Sønderby, D. Nouri, D. Maturana, M. Thoma, E. Battenberg, J. Kelly, *et al.*, "Lasagne: first release," *Zenodo: Geneva, Switzerland*, vol. 3, 2015.

[19] K. Banker, *MongoDB in action*. Manning Publications Co., 2011.

[20] K. Chodorow, *MongoDB: The Definitive Guide: Powerful and Scalable Data Storage*. .ºReilly Media, Inc.", 2013.

[21] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *nature*, vol. 521, no. 7553, p. 436, 2015.

[22] H. Li, Z. Lin, X. Shen, J. Brandt, and G. Hua, "A convolutional neural network cascade for face detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 5325–5334, 2015.

[23] R. Ranjan, S. Sankaranarayanan, C. D. Castillo, and R. Chellappa, "An all-in-one convolutional neural network for face analysis," in *Automatic Face & Gesture Recognition (FG 2017), 2017 12th IEEE International Conference on*, pp. 17–24, IEEE, 2017.

[24] D. Shen, G. Wu, and H.-I. Suk, "Deep learning in medical image analysis," *Annual review of biomedical engineering*, vol. 19, pp. 221–248, 2017.

[25] M. Bojarski, D. Del Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang, *et al.*, "End to end learning for self-driving cars," *arXiv preprint arXiv:1604.07316*, 2016.

[26] I. Heredia, "Large-scale plant classification with deep neural networks," in *Proceedings of the Computing Frontiers Conference*, pp. 259–262, ACM, 2017.

[27] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, pp. 1097–1105, 2012.

[28] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[29] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. A. Alemi, "Inception-v4, inception-resnet and the impact of residual connections on learning.," in *AAAI*, vol. 4, p. 12, 2017.

[30] S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He, "Aggregated residual transformations for deep neural networks," in *Computer Vision and Pattern Recognition (CVPR), 2017 IEEE Conference on*, pp. 5987–5995, IEEE, 2017.

[31] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *Proceedings of the 27th international conference on machine learning (ICML-10)*, pp. 807–814, 2010.

[32] X. Glorot, A. Bordes, and Y. Bengio, "Deep sparse rectifier neural networks," in *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pp. 315–323, 2011.

[33] C. M. Bishop, *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Berlin, Heidelberg: Springer-Verlag, 2006.

[34] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pp. 248–255, Ieee, 2009.

[35] K. P. Murphy, *Machine Learning: A Probabilistic Perspective*. The MIT Press, 2012.

[36] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[37] S. Ruder, "An overview of gradient descent optimization algorithms," *arXiv preprint arXiv:1609.04747*, 2016.

[38] N. Qian, "On the momentum term in gradient descent learning algorithms," *Neural networks*, vol. 12, no. 1, pp. 145–151, 1999.

[39] T. Tieleman and G. Hinton, "Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude," *COURSERA: Neural networks for machine learning*, vol. 4, no. 2, pp. 26–31, 2012.

[40] S. C. Wong, A. Gatt, V. Stamatescu, and M. D. McDonnell, "Understanding data augmentation for classification: when to warp?," *arXiv preprint arXiv:1609.08764*, 2016.

[41] M. Sokolova and G. Lapalme, "A systematic analysis of performance measures for classification tasks," *Information Processing & Management*, vol. 45, no. 4, pp. 427–437, 2009.

[42] H. Zheng, R. Wang, Z. Yu, N. Wang, Z. Gu, and B. Zheng, "Automatic plankton image classification combining multiple view features via multiple kernel learning," *BMC bioinformatics*, vol. 18, no. 16, p. 570, 2017.

[43] C. Cortes and V. Vapnik, "Support-vector networks," *Machine learning*, vol. 20, no. 3, pp. 273–297, 1995.

[44] M. Gönen and E. Alpaydın, "Multiple kernel learning algorithms," *Journal of machine learning research*, vol. 12, no. Jul, pp. 2211–2268, 2011.

[45] H. M. Sosik and R. J. Olson, "Automated taxonomic classification of phytoplankton sampled with imaging-in-flow cytometry," *Limnology and Oceanography: Methods*, vol. 5, no. 6, pp. 204–216, 2007.

[46] G. Gorsky, M. D. Ohman, M. Picheral, S. Gasparini, L. Stemmann, J.-B. Romagnan, A. Cawood, S. Pesant, C. García-Comas, and F. Prejger, "Digital zooplankton image analysis using the zooscan integrated system," *Journal of plankton research*, vol. 32, no. 3, pp. 285–303, 2010.

[47] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "Smote: synthetic minority over-sampling technique," *Journal of artificial intelligence research*, vol. 16, pp. 321–357, 2002.

[48] S. García and F. Herrera, "Evolutionary undersampling for classification with imbalanced datasets: Proposals and taxonomy," *Evolutionary computation*, vol. 17, no. 3, pp. 275–306, 2009.